Amplicon sequencing (16S and ITS) processing platform

Thomas Gurry (thomasgurry@gmail.com) and Claire Duvallet (duvallet@mit.edu)

Center for Microbiome Informatics and Therapeutics

Contents

1 Introduction

This document describes the process of going from raw 16S or ITS data to processed data (OTU tables, oligotypes, etc.). This is orchestrated by the script `raw2otu.py`. The platform is designed in such a manner that the user interfaces with a single script, `Master.py`, which takes as input the path to the folder containing the dataset. Each dataset folder must contain a machine-readable text file, called a summary file, which gives instructions to `Master.py`. The format of these summary files, and all files required for 16S or ITS processing, are described in the section below.

2 Preparing your dataset for processing

2.1 Raw data files

The first step to process your data is to understand what raw data you have. Do you have FASTQ or FASTA data? Are your sequences already de-multiplexed with one file per sample, or will you need to split sequences by barcodes? Do you have unmerged forward and reverse reads that you'll need to merge? Are the primers still in the sequences? Are

they already all trimmed to a certain length? Every sequencing center provides a different kind of "raw data", so make sure you know what you're starting with! The pipeline takes many different kinds of inputs and can perform many different processing steps, so it's important to know what you'll be needing.

2.2 Summary file

This file, named `summary_file.txt`, is a machine-readable, **tab-delimited** file that must accompany any dataset directory when uploaded to the cloud. It orchestrates all processing that will happen, and is where you specify each of your processing requests. It should be found in the highest directory level for the dataset directory in the S3 bucket. It is a text file with descriptors for the data and paths to all relevant datafiles within the directory. It can include a True or False flag for whether any associated raw 16S/ITS data has already been processed. It is case-sensitive.

The order in which items are listed between the lines `#16S_start` and `#16S_end` (for 16S) and between lines `#ITS_start` and `#ITS_end` (for ITS) does not matter.

Note that any white space in the summary file should correspond to a single tab character.

2.3 Summary file format and options

The first line in your summary file should be:

```
DATASET_ID myDataset
```

16S and ITS processing parameters are specified by attributes placed on separate lines between `#16S_start` and `#16S_end` and/or `#ITS_start` and `#ITS_end`. All white spaces in the summary file should be tabs.

Required attributes are: `PRIMERS_FILE`, `BARCODES_MAP`, `PROCESSED` and one of: `RAW_FASTQ_FILES`, `RAW_FASTA_FILE`, and `RAW_FASTA_FILES`.

For processing to occur, `PROCESSED` should be `False` (case-sensitive).

The following section will go through all available summary file attributes, and are summarized in Appendix 8.0.1. These options are presented in roughly the same order as they are processed.

2.3.1 Input file(s)

The pipeline takes as input many kinds of raw data:

- one FASTQ file. This file should contain all sequences for all of your samples. Sequences may still contain barcodes, or they can have had barcodes removed and FASTQ headers replaced with sample identifiers.

- multiple FASTQ files. Each of these files should contain sequences for one sample only.
- one FASTA file. This file should contain all sequences for all of your samples, and should have a sample identifier in the FASTA header line.
- multiple FASTA files. Each of these files should contain sequences for one sample only.

You can specify the type of file input by using the , `RAW_FASTQ_FILES`, `RAW_FASTA_FILE`, and `RAW_FASTA_FILES` attributes.

If you have one sequence file, `RAW_FASTQ_FILE` or `RAW_FASTA_FILE` should refer to the FASTQ/A file name. If you have multiple sequence files, `RAW_FASTQ_FILES` or `RAW_FASTA_FILES` should refer to a text file that relates each FASTQ/A file to its sample identifier. This text file is tab-delimited with the FASTQ/A file name in the first column and the corresponding sample ID in the second column. This text file should not contain a header.

Note that all file paths provided should be *relative to the summary file directory.* If your FASTQ to sample ID file map is in a subdirectory called `filemaps/` and your sequence files are in a subdirectory called `datafiles/`, then `RAW_FASTQ_FILES` would be `filemaps/fastq_filemap.txt`, and `fastq_filemap.txt` would include `filemaps/file1.fastq`, etc.

2.3.2 Merging

If you have unmerged paired-end reads, you can merge them by setting `MERGE_PAIRS` to `True` (case sensitive). Merging can be performed in both the case where your reads are not de-multiplexed (i.e. you have one file with *all* of your forward reads for all of your samples, and one file with all of the reverse reads) and when they are (i.e. you have two files per sample: one with the forward reads and one with the reverse reads).

If you need to merge reads, you also need to specify the file name suffixes for both the forward and reverse read files. These are specified in `FWD_SUFFIX` and `REV_SUFFIX` and default to `_R1.fastq` and `_R2.fastq`, respectively. If you have more complicated file names, either rename them to have consistent suffixes, or talk to Claire to see if you can incorporate more complicated regex matching into the code.

Note that the files specified either in `RAW_FASTQ_FILE` or in the `fastq_filemap.txt` (specified in `RAW_FASTQ_FILES`) should refer to the FASTQ files containing the forward reads.

See Case 4 in Section 2.4.4 for an example.

2.3.3 De-multiplexing

If your FASTQ file still contains the barcodes in the sequences, you will need to include `BARCODES_MAP` and `BARCODES_MODE` in your summary file. ****Note that** `BARCODES_MAP` **is**

a required attribute. If you do not need to de-multiplex your sequences, `BARCODES_MAP` should be `None` (case-sensitive).

`BARCODES_MAP` refers to a tab-delimited file which has the sample identifiers in the first column and the corresponding barcode sequence in the second column. This file does not have a header.

You must also specify a `BARCODES_MODE`, which specifies where the barcodes are to found in the FASTQ file. If the barcodes are still in the sequences, `BARCODES_MODE` should be 2. If the barcodes are in the FASTQ sequence header, `BARCODES_MODE` should be 1.

See Case 1 in Section 2.4.1 for an example.

Sometimes, the 'raw' data has already had primers and barcodes removed but still has all samples in the same FASTQ file. In this case, `BARCODES_MAP` should be `None` and the sample IDs must be listed in the sequence header lines of the FASTQ file. If there is text other than the sample ID in the header, you need to specify the first non-sample ID character in `BARCODES_SEPARATOR`. For example, sequences in these kinds of files are often labeled like:

```
@sample1_seq1
...<rest of fastq record>
@sample1_seq2
...<rest of fastq record>
@sample2@_seq1
...<rest of fastq record>
@sample3_seq1
...<rest of fastq record>
@sample2_seq2
...<rest of fastq record>
```

In this case, the barcodes separator would be an underscore (_), which is the default.

2.3.4 Primer trimming

If you need to remove primers from your sequences, you can specify `PRIMERS_FILE`, a text file with your primer sequences. ****Note that `PRIMERS_FILE` is a required attribute.** If you do not need to remove primers from your sequences, `PRIMERS_FILE` should be `None` (case sensitive).

The pipeline does not currently remove reverse primers. If your sequences still contain reverse primers, you can remove them yourself or trim your sequences to a length shorter than the start of your reverse primer.

2.3.5 Quality filtering

There are two ways to quality filter your sequences. One is based on the number of expected errors in your sequence, and the other truncates reads after a certain quality

is encountered. You can learn more about these approaches by reading the USEARCH documentation: http://www.drive5.com/usearch/manual/readqualfiltering.html

To truncate reads after a base with a certain quality is encountered, use the `QUALITY_TRIM` option. A default value that is often used is 25. This step is performed *before* length trimming.

To discard reads based on their number of expected errors, use the `MAX_ERRORS` option. A default value that is often used is 2 (i.e. reads with more than 2 expected errors are discarded). This step is performed *after* length trimming

If nothing is specified, the pipeline defaults to `QUALITY_TRIM` of 25. If both `MAX_ERRORS` and `QUALITY_TRIM` are specified, quality filtering by truncation is performed (i.e. `MAX_ERRORS` is ignored).

You may also need to specify the encoding of the quality scores. `ASCII_ENCODING` can be either `ASCII_BASE_33` (default) or `ASCII_BASE_64`. You can check the encoding of your file using usearch: `usearch -fastq_chars yourFASTQfile.fastq`

2.3.6 Length trimming

By default, the pipeline trims all reads to 101 base pairs before dereplication and clustering. You can specify a different length by using `TRIM_LENGTH`. Any reads which are shorter than the specified length are discarded.

2.3.7 Dereplication

In the dereplication step, unique sequences are identified and the samples from which they came are tracked (sometimes referred to as "provenancing"). By default, unique sequences which are present fewer than 10 times in the entire dataset are discarded. If you want to change this number, specify it with `MIN_COUNT`. (e.g. if `MIN_COUNT` is 2, only singleton sequences are discarded).

2.3.8 OTU calling

You can specify the similarity used to define OTUs in the `OTU_SIMILARITY` attribute. The default value is 97, corresponding to 97% OTUs.

By default, the pipeline clusters OTUs using both *de novo* and closed-reference approaches. If you specify an OTU similarity that does not have a corresponding Green Genes reference file, closed-reference clustering will not be performed. OTU similarities supported by Green Genes closed-reference mapping are: 61, 64, 67, 70, 73, 76, 79, 82, 85, 88, 91, 94, 97, and 99%. The database files used for this mapping can be found in `/home/ubuntu/databases/gg_13_5_otus/rep_set_latin/`.

The pipeline assigns taxonomies to *de novo* OTUs using the naive-Bayes RDP classifier. By default, the confidence cutoff is 0.5. You can specify a different value with the `RDP_CUTOFF` attribute.

2.3.9 Distribution-based OTU calling

The pipeline also performs distribution-based OTU calling[1]. You can set the abundance, distance and p value criteria in the summary file attributes `DISTANCE_CRITERIA`, `ABUNDANCE_CRITERIA`, and `DBOTU_PVAL`.

Distribution-based clustering is performed by default, in addition to the *denovo* and closed-reference methods. If you find that it's breaking or it takes too long, you can turn it off by setting the summary file attribute `DBOTU` to `False`.

2.4 Sample summary files

2.4.1 Case 1: raw FASTQ file of 16S sequences, still includes primers and barcodes

The simplest case is if you have the following files: a raw FASTQ file; a file specifying the map between barcode sequences and IDs; and a file specifying the primers used. Your summary file would look something like this:

```
DATASET_ID myDataset

#16S_start
RAW_FASTQ_FILE      myData.fastq
ASCII_ENCODING      ASCII_BASE_33
PRIMERS_FILE        primers.txt
BARCODES_MAP        barcodes_map.txt
BARCODES_MODE       2
METADATA_FILE       metadata.txt
PROCESSED           False
#16S_end
```

Note that you **must** also specify the place where barcodes are to be found, i.e. either in the ">" sequence ID lines (mode 1) or in the sequences themselves (mode 2). The `PROCESSED` flag tells the processing instance that the dataset needs to be processed into OTU tables.

Your `barcodes_map.txt` file would look something like this:

```
S1      ATCGCTAGTA
S2      TCGCTATATA
S3      TCTACAGCGT
S4      CGTACTCAGT
```

[1] http://almlab.mit.edu/dbotu3.html

2.4.2 Case 2: raw FASTQ file of ITS sequences, primers and barcodes have been removed

In the case where the 'raw' data has already had primers and barcodes removed (but is not yet de-multiplexed, i.e. all samples are still in the same FASTQ file), the sample IDs must be listed in the sequence ID lines of the FASTQ file. When the pipeline removes barcodes itself and replaces them with sample IDs, individual sequence reads for a given `sampleID` will be annotated as `sampleID;1, sampleID;2`, etc., where we note here that the `BARCODES_SEPARATOR` is ';'. However, in a dataset where the barcodes have previously been removed, you will have to look into the FASTQ file to check the 'separator' character. Your summary file would look something like this:

```
DATASET_ID myDataset

#ITS_start
RAW_FASTQ_FILE     myData.fastq
ASCII_ENCODING     ASCII_BASE_33
PRIMERS_FILE       None
BARCODES_MAP       None
BARCODES_SEPARATOR ;
METADATA_FILE      metadata.txt
PROCESSED          False
#ITS_end
```

2.4.3 Case 3: multiple demultiplexed raw FASTQ or FASTA files of 16S sequences, each file corresponding to a single sample

Sometimes sequencing data are available in a demultiplexed form, where the reads for each sample are split into separate files. Many datasets in the SRA, for example, are available in this form. In this case, you can create a **two-column, tab-delimited file** where the first column lists the filename and the second column lists the corresponding sample ID. Note that paths should be relative paths *within the current directory*, e.g. `datafiles/file1.txt` for a folder called `datafiles` within the current directory. In the summary file, the `RAW_FASTQ_FILE` line becomes `RAW_FASTQ_FILES` (plural), and instead refers to this filename. If your files are FASTA rather than FASTQ, simply use `RAW_FASTA_FILES` (also plural). For a filename `fastq_filemap.txt`, your summary file would look something like this:

```
DATASET_ID myDataset

#16S_start
RAW_FASTQ_FILES     fastq_filemap.txt
```

```
ASCII_ENCODING      ASCII_BASE_33
PRIMERS_FILE        primers.txt
METADATA_FILE       metadata.txt
PROCESSED           False
PRIMERS_FILE        None
BARCODES_MAP        None
#16S_end
```

And your `fastq_filemap.txt` file would look something like this (note that white spaces in the following example correspond to a single tab character):

```
SRR10001.fastq S1
SRR10002.fastq S2
SRR10003.fastq S3
SRR10004.fastq S4
```

2.4.4 Case 4: multiple demultiplexed raw paired-end FASTQ files of 16S sequences which need merging

If your 16S FASTQ files are split into forward and reverse paired-end reads, the pipeline can merge them for you. Specify `MERGE_PAIRS` in the summary file, and also include the filename suffixes corresponding to forward and reverse reads. If your forward read fastq files were named `sampleID_L001_R1.fastq` and your reverse read fastq files were named `sampleID_L001_R2.fastq`, your summary file would look something like this:

```
DATASET_ID myDataset

#16S_start
RAW_FASTQ_FILES fastq_filemap.txt
PRIMERS_FILE    None
BARCODES_MAP    None
MERGE_PAIRS     True
FWD_SUFFIX      _L001_R1.fastq
REV_SUFFIX      _L001_R2.fastq
PROCESSED       False
#16S_end
```

If you have de-multiplexed files (as in this example), the file names in your `fastq_filemap.txt` file should be the forward read fastq files.

If instead you have non-demultiplexed sequences (i.e. two fastq files, one containing your forward reads and one containing your reverse reads), `RAW_FASTQ_FILE` should point to the file containing the forward reads.

3 Running the pipeline

3.1 Processing your data

Once you have placed all relevant files and folders, including the summary file, into a single folder, you can call the script `Master.py` with the path to this folder as input. `Master.py` will parse the summary file and launch the appropriate scripts to process your request. Suppose the path to the folder containing the dataset is `/home/ubuntu/dataset_folder`, and contains a summary file where you specified the dataset ID as 'myDataset', you would run the following command from anywhere:

```
python ~/scripts/Master.py -i /home/ubuntu/dataset_folder
```

If you prefer to run the program in the background, you can add an `&` to the end of the command. Alternatively, you can use your favorite terminal multi-plexer (e.g. screen or tmux) to run the code while continuing your work. Using `nohup` ensures that any hangups do not interrupt the processing:

```
nohup python ~/scripts/Master.py -i /home/ubuntu/dataset_folder &
```

Processing happens in a folder within the `proc` folder, with path `/home/ubuntu/proc/myDataset_proc_16S` or `/home/ubuntu/proc/myDataset_proc_ITS`, depending on the amplicon being analyzed. Final results are put in `/home/ubuntu/processing_results/`. Log files are put in the `/home/ubuntu/logs` folder, with two files created for each dataset: `stderr_datasetID.log` for error and warning messages and `stdout_datasetID.log` for various progress messages. Certain std out and std err messages also go to the `stderr_master.log` and `stdout_master.log` in the directory from which you called the `python Master.py` command. Make sure to check both of these places when troubleshooting!

Please remove your files from the ~/proc folder once you have checked your processing results! The ~/proc folder can easily fill up - it contains the original raw files are and additional copies corresponding to each processing step!

3.2 Underlying function calls

The following table describes the underlying scripts and functions used in each step of 16S and ITS data processing. Python scripts are passed-down Alm lab scripts (for the most part).

All code is in `/home/ubuntu/scripts` and we encourage users to take a look!

Processing	**Function/script call**
Merging	`usearch8 -fastq_mergepairs`

De-multiplexing	`2.split_by_barcodes.py`
Primer trimming	`1.remove_primers.py`
Quality filtering (truncation)	`usearch8 -fastq_filter -fastq_truncqual`
Quality filtering (expected errors)	`usearch8 -fastq_filter -fastq_maxee`
Length trimming (FASTQ)	`usearch8 -fastq_filter -fastq_trunclen`
Length trimming (FASTA)	`usearch8 -fastx_truncate -trunclen`
Dereplication	`3.dereplicate.py`
De novo clustering	`usearch8 -cluster_otus -otu_radius_pct`
Closed-reference mapping	`usearch8 -usearch_global -db GG_database_file -strand both`
RDP taxonomy assignment	`rdp_classify.py`
Distribution-based clustering	`dbotu.py call_otus()`

4 Pipeline output files and directories

The pipeline outputs different OTU tables and corresponding representative sequences. All final outputs can be found in the `processing_results` folder, under a sub-directory labeled `myDataset_results`. Files in this directory are labeled systematically, usually with the format `myDataset.file_description.otu_similarity.file_type`.

4.1 Top directory

Files in the top results directory are as follows:

- `myDataset.otu_seqs.N.fasta`: FASTA file with the representative sequences for the denovo OTUs, clustered at N%.
- `myDataset.otu_seqs.dbOTU.fasta`: FASTA file with the representative sequences for the distribution-based OTUs.
- `myDataset.otu_table.N.denovo`: OTU table with N% denovo OTUs labeled `denovo1`, `denovo2`, ... in the rows and samples in the columns.
- `myDataset.otu_table.N.dbOTU`: OTU table with distribution-based OTUs labeled `dbotu1`, `dbotu2`, ... in the rows and samples in the columns.

- `myDataset.otu_table.N.denovo_oligotypes`: OTU table with N% denovo OTUs separated into unique oligotypes. Each OTU is labeled `denovo1.1`, `denovo1.2`, `denovo2.1`, `denovo2.2`, `denovo2.3`, etc. The first number corresponds to the parent denovo OTU number; the second is the oligotype number. Oligotypes are calculated as each unique sequence within an OTU cluster.
- `myDataset.raw_dereplicated.fasta`: FASTA file with all unique sequences in the dataset. Only sequences which appear more times than the `MIN_COUNT` specified in the summary file are included (default value for `MIN_COUNT` is 10).
- `summary_file.txt`: Updated summary file containing original processing request and resulting file names.

Also within each dataset's `processing_results` directory, 3 sub-directories are created:

4.2 Quality control

Within the results folder, there is a subfolder called `quality_control`, which contains various plots diagnostic of dataset quality. Currently, the pipeline outputs:

- Histogram showing distribution of read lengths, taken from the first 100,000 reads in the raw FASTQ file.
- Bar chart showing number of reads per sample.
- File showing percentage of reads thrown out at each processing step (`processing_summary.txt`).

Note that the information in these files is not always accurate - you should probably do more thorough quality control yourself.

4.3 RDP

This folder contains the RDP-assigned OTU tables. It has two files:

- `myDataset.otu_table.N.denovo.rdp_assigned`: OTU table with denovo OTUs assigned Latin names with RDP. OTUs are in rows and samples are in columns. OTU names are of the format:

  ```
  k__kingdom;p__phylum;c__class;o__order;f__family;g__genus;s__species;
  d__denovoID
  ```

  where the denovoID corresponds to the respective sequence in `../myDataset.otu_seqs.N.fasta`.
- `myDataset.otu_table.dbOTU.rdp_assigned`: OTU table with distribution-based OTUs assigned Latin names with RDP. OTUs are in rows and samples are in columns. OTU names are of the format:

```
k__kingdom;p__phylum;c__class;o__order;f__family;g__genus;s__species;
d__dbotuID
```

where the dbotuID corresponds to the respective sequence in `../myDataset.otu_seqs.dbOTU.fasta`.

4.4 GG

This folder contains the Green Genes-assigned OTU table. It has multiple files, which are described further in the following section.

- myDataset.otu_table.N.gg.consensusM: closed-reference OTU table with dereplicated sequences mapped to Green Genes using `usearch`. OTUs are in rows and samples are in columns. OTU names are of the format:

  ```
  k__kingdom;p__phylum;c__class;o__order;f__family;g__genus;s__species;
  d__derepID
  ```

  where the derepID corresponds to the sequence in `../myDataset.raw_dereplicated.fasta` with the same ID number.

5 Description of pipeline OTU tables

5.1 OTU calling

The pipeline produces various OTU tables. All tables are constructed from the set of raw dereplicated reads. These are a set of reads in FASTA format that appear in the file `datasetID.raw_dereplicated.fasta` and correspond to the set of unique sequences present in the raw data.

5.1.1 *de novo* OTU tables

The dereplicated reads are clustered to within the specified sequence similarity percentage cut-off (`OTU_SIMILARITY`, default: 97) using `usearch`. Individual reads are assigned either as the OTU centroid or as a match. Centroids are labeled as 'OTU_ID.0' and matches count from 'OTU_ID.1' onwards. These are separate oligotypes within the OTU 'OTU_ID'. All oligotype counts are then collapsed to their respective OTU, resulting in a fully *de novo* OTU table which can be found in the filename `datasetID.otu_table.*.denovo` where '`*`' gets replaced by the OTU similarity cut-off.

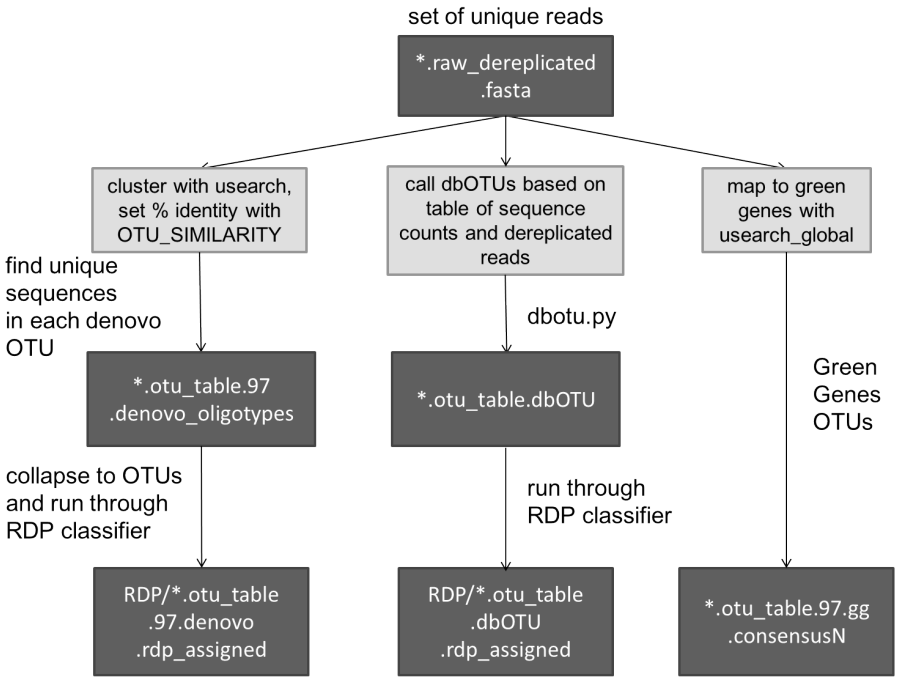

Figure 1: Schematic of OTU calling pipeline.

5.1.2 Distribution-based OTU tables

A counts table is made from the dereplicated reads and dereplication map. The counts table and dereplicated reads are used as inputs to `dbotu.py`'s `call_otus()` function. Briefly, sequences are compared against existing OTU representative sequences in descending abundance order. If the candidate sequence's abundance is within the `ABUNDANCE_CRITERIA` of an existing OTU and its count distribution across samples is not significantly different from the OTU's, then the candidate sequence is grouped with that OTU. You can learn more about the distribution-based clustering algorithm at `http://almlab.mit.edu/dbotu3.html`.

Intermediate files such as the membership file (which lists which sequences belong to which OTU) and the table of sequence counts (used as an input to the `dbotu.py` function call) are in the respective `/home/ubuntu/proc/` folder for your dataset.

5.1.3 Closed-reference OTU table

There are two types of database-referencing that are outputed from the pipeline by default. The first is assignments from the Ribosomal Database Project (RDP), which returns probabilistic assignments. The default probability cut-off below which an assignment is labeled as unidentified at a given taxonomic level is 0.5, but this can be set using a summary file parameter `RDP_CUTOFF`. This OTU table can be found in the results sub-folder called 'RDP'. The full classifications can be found in `/home/ubuntu/proc/`, in `RDP_classifications.denovo.txt` or `RDP_classifications.dbOTU.txt`.

The dereplicated reads are also aligned to a standard database (GreenGenes in the case of 16S sequences and UNITE in the case of ITS sequences). In the case of GreenGenes, the database is determined based on the specified OTU similarity cut-off: e.g. `97_otus.fasta` for `OTU_SIMILARITY` set to 97). The alignment is performed using `usearch`, and considers the top 10 hits. Consensus assignments are then produced for the top 1, top 3, top 5 and top 10 hits (where a taxonomic level is only assigned a latin name if the top N hits from GreenGenes agree), and the corresponding OTU tables are output. Thus, the OTU table called `datasetID.otu_table.*.gg.consensus10` contains latin names which are formed from a minimum consensus of the top 10 hits for each taxonomic level, where '`*`' gets replaced by the OTU similarity cut-off. Levels are left unidentified (e.g. '`s__`') if the consensus requirement is not met.

Note that the database referencing process is one of the slower steps in the pipeline, so if you only care about RDP, you can skip the GreenGenes assignments steps by setting the parameter `GG_ALIGN` to `False` in the summary file. Similarly, for ITS sequences, you can skip the UNITE assignments steps by setting the parameter `UNITE_ALIGN` to `False`.

5.1.4 Open-reference OTU table

`DEPRECATED:` The pipeline no longer produces open-reference OTUs.

If any reads in the dereplicated sequences do not align to GreenGenes/UNITE to within the specified similarity cut-off, they are clustered with the desired similarity cut-off using `usearch`, and appended to the GreenGenes/UNITE closed-reference table to produce a full, open-reference OTU table (which combines GreenGenes-referenced and *de novo* OTUs) at the desired similarity cut-off, with filename `datasetID.otu_table.gg.*.open_ref`.

5.1.5 Oligotypes

The OTU calling process automatically assigns separate IDs to unique sequences within OTUs (termed 'oligotypes' by some). These offer a greater level of granularity. For example, if sequences A, B and C are all the same OTU, called OTU1 (and the centroid is sequence A), these would be relabeled 'OTU1.0', 'OTU1.1' and 'OTU1.2'. Note that these unique 'oligotypes' are generated by any fancy information-based algorithms - they are simply the unique sequences within each OTU.

There is currently one such table output: the full *de novo* oligotype table (where sequences are labeled according to a fully *de novo* OTU clustering process) - `myDataset.otu_table.N.denovo_oligotypes`

6 Source code

The following code can all be in `/home/ubuntu/scripts`. Some of these scripts are used by the pipeline, others are not currently incorporated but may still prove useful to your 16S-related work.

6.1 Python modules

- `Formatting.py` - Module to house miscellaneous formatting methods, e.g. conversion from classic dense format to BIOM format, OTU table transposition, etc.
- `QualityControl.py` - Methods for quality control diagnostics on a dataset.
- `preprocessing_16S.py` - Methods and wrappers for raw 16S sequence data processing.
- `Taxonomy.py` - Methods for taxonomy-related feature extraction and analytics. Includes functions for things like: -adding latin names to a GreenGenes-referenced OTU table -collapsing abundances at different taxonomic levels
- `Phylogeny.py` - Methods for phylogenetic feature extraction, e.g. left/right (LR) abundance ratios at each node of a phylogenetic tree.
- `Analytics.py` - Generic statistical analysis tools, e.g. Wilcoxon tests across all available taxa.

- `Regressions.py` - Performs different types of regressions *en masse*.
- `PipelineFilesInterface.py` - Methods for reading and moving around raw data files and for reading specific groups of attributes from the summary file.

6.2 Scripts and routines

- `Master.py` - Master script that calls relevant processing pipelines, e.g. `raw2otu.py`.
- `raw2otu.py` - Pipeline for converting raw 16S FASTQ sequence files to OTU tables. Handles parallelization requirements in these processing steps automatically. Takes as input a directory that contains a summary file and the raw data.
- `dbotu.py` - Module for doing distribution-based OTU calling.
- `rdp_classify.py` - Wrapper for calling the RDP classifier jar file, found at `/home/ubuntu/tools/RDPTools/classifier.jar`.

7 Troubleshooting

Standard error (`stderr`) and standard out (`stdout`) are directed to the directory `/home/ubuntu/logs`. Errors will appear in the `stderr` file for the corresponding datasets. There may also be useful errors in the `stderr_master.log` and `stdout_master.log` files that are created in the directory from which you ran the `python Master.py` call. In addition, file outputs from each step of the pipeline can be found in `/home/ubuntu/proc/datasetID_proc_16S`, which can help to diagnose errors. Common sources of errors or difficulties include:

- Bad formatting of the summary, barcodes, or primers file. There should be no white spaces, and columns should be tab-delimited. If you created the file in Excel, it may have carriage return or other non-linux formatting characters that introduce difficulties. Safest is to go from a previous summary file template, or to create it manually.
- Typos in a 16S attribute key-value pair in the summary file.
- Incorrect ASCII encoding specified. Check whether 33 or 64 using `usearch8 -fastq_chars yourFASTQfile.fastq`. Default is 33, so if left unspecified and the file is ASCII base 64, quality trimming will fail.
- For ITS sequences, the RDP classifier often runs out of RAM when loading the UNITE database. This is usually what happened if you find `RDP_classification.txt` to be empty in the relevant folder in `/home/ubuntu/proc`. If you rerun the pipeline several times, have checked everything else and keep encountering this problem, you may need a node with more RAM to do your processing.

- If you specify a file containing a list of FASTQ/FASTA files for the raw reads, make sure the file paths are relative to the directory hosting the summary file.

8 Appendix

8.0.1 List of 16S and ITS attributes

Attribute	Description
RAW_FASTQ_FILE	Raw FASTQ file name/path within the dataset directory
RAW_FASTA_FILE	Raw FASTA file name/path if raw data is in FASTA format
RAW_FASTQ_FILES	For demultiplexed datasets where samples are separated into separate FASTQ files. Filename of two column file containing FASTQ filenames in first column and sample IDs in the second column.
RAW_FASTA_FILES	For demultiplexed datasets where samples are separated into separate FASTA files. Filename of two column file containing FASTA filenames in first column and sample IDs in the second column.
ASCII_ENCODING	ASCII quality encoding in FASTQ. Supports either 'ASCII_BASE_33' or 'ASCII_BASE_64'. Set to 33 if unspecified.
PRIMERS_FILE	Filename/path to primers file. `required`: If primers have already been removed, specify 'None'.
BARCODES_MAP	Filename/path to barcodes map file. Tab-delimited file contains sampleIDs in first column and barcode sequences in second column. `required`: If barcodes have already been removed, specify 'None'.
BARCODES_MODE	'1' = barcodes in sequence ID, '2' = barcodes in sequences themselves. Required if `BARCODES_MAP` is not None.
BARCODES_SEPARATOR	Separator character. See description in Case 2 below.
METADATA_FILE	Filename/path to metadata file.
MERGE_PAIRS	If need to merge paired-end reads, set to "True".
FWD_SUFFIX	Filename suffix of files with forward reads. Should include filename extension. If not specified, defaults to `_1.fastq`
REV_SUFFIX	Filename suffix of files with reverse reads. Should include filename extension. If not specified, defaults to `_2.fastq`
PROCESSED	True/False flag for whether data have already been processed. `required`: Set to 'False' for processing to proceed.
TRIM_LENGTH	Length to which all sequences should be trimmed. Defaults to 101 if unspecified.

QUALITY_TRIM	Minimum quality score allowed. Sequences are truncated at the first base having quality score less than value. Defaults to 25 if unspecified. If set to None, no quality filtering or trimming will be performed. If both QUALITY_TRIM and MAX_ERRORS are included in summary file, MAX_ERRORS will be ignored (even if QUALITY_TRIM = None).
MAX_ERRORS	Maximum expected errors allowed. After length trimming, sequences with more than MAX_ERRORS expected errors are discarded. If not specified or if a TRIM_QUALITY value is specified, defaults to quality trimming behavior, above.
MIN_COUNT	Minimum sequence count in dereplication across all samples. Defaults to 10 if unspecified (i.e. sequences with fewer than 10 occurrences in the entire dataset will not be considered downstream).
OTU_SIMILARITY	Integer specifying the percent similarity desired in OTU clustering. Defaults to 97 if unspecified.
RDP_CUTOFF	Desired probability cut-off for Ribosomal Database Project assignments. Assignments at each taxonomic level will be evaluated and those with a lower probability than this cutoff will be labeled as unidentified. Defaults to 0.5 if unspecified.
GG_ALIGN	Specific to 16S sequences. True/False flag for whether GreenGenes alignments are desired. Defaults to 'True' if unspecified.
UNITE_ALIGN	Specific to ITS sequences. True/False flag for whether UNITE alignments are desired. Defaults to 'True' if unspecified.
DBOTU	Whether to perform distribution-based OTU calling. Defaults to True if unspecified.
ABUNDANCE_CRITERIA	Abundance criteria for distribution-based OTU calling. Defaults to 10 if unspecified.
DISTANCE_CRITERIA	Distance criteria for distribution-based OTU calling. Defaults to 0.1 if unspecified.
DBOTU_PVAL	P value cutoff for distribution-based OTU calling. Defaults to 0.0005 if unspecified.
OUTDIR	Full path to processing directory. Defaults to /home/ubuntu/proc/ if not specified.